FORTL

```
FFFFFFFFF    000000    RRRRRRRR   EEEEEEEEEE  RRRRRRRR   RRRRRRRR    000000    RRRRRRRR
FFFFFFFFF    000000    RRRRRRRR   EEEEEEEEEE  RRRRRRRR   RRRRRRRR    000000    RRRRRRRR
FF          00    00   RR    RR   EE          RR    RR   RR    RR   00    00   RR    RR
FF          00    00   RR    RR   EE          RR    RR   RR    RR   00    00   RR    RR
FF          00    00   RR    RR   EE          RR    RR   RR    RR   00    00   RR    RR
FF          00    00   RR    RR   EE          RR    RR   RR    RR   00    00   RR    RR
FFFFFFFF    00    00   RRRRRRRR   EEEEEEE     RRRRRRRR   RRRRRRRR   00    00   RRRRRRRR
FFFFFFFF    00    00   RRRRRRRR   EEEEEEE     RRRRRRRR   RRRRRRRR   00    00   RRRRRRRR
FF          00    00   RR  RR     EE          RR  RR     RR  RR     00    00   RR  RR
FF          00    00   RR  RR     EE          RR  RR     RR  RR     00    00   RR  RR
FF          00    00   RR    RR   EE          RR    RR   RR    RR   00    00   RR    RR
FF          00    00   RR    RR   EE          RR    RR   RR    RR   00    00   RR    RR
FF           000000    RR    RR   EEEEEEEEEE  RR    RR   RR    RR    000000    RR    RR   ....
FF           000000    RR    RR   EEEEEEEEEE  RR    RR   RR    RR    000000    RR    RR   ....


LL           IIIIII    SSSSSSSS
LL           IIIIII    SSSSSSSS
LL             II     SS
LL             II     SS
LL             II     SS
LL             II     SS
LL             II      SSSSSS
LL             II      SSSSSS
LL             II           SS
LL             II           SS
LL             II           SS
LL             II           SS
LLLLLLLLLL   IIIIII    SSSSSSSS
LLLLLLLLLL   IIIIII    SSSSSSSS
```

```
   1    0001   0 MODULE FOR$$ERROR (%TITLE 'Internal FORTRAN error handling module'
   2    0002   0                  IDENT = '1-022'            ! File: FORERROR.B32  Edit: SBL1022
   3    0003   0                  ) =
   4    0004   1 BEGIN
   5    0005   1 !
   6    0006   1 !*********************************************************************
   7    0007   1 !*                                                                   *
   8    0008   1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                          *
   9    0009   1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.           *
  10    0010   1 !*  ALL RIGHTS RESERVED.                                             *
  11    0011   1 !*                                                                   *
  12    0012   1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
  13    0013   1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
  14    0014   1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
  15    0015   1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
  16    0016   1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
  17    0017   1 !*  TRANSFERRED.                                                      *
  18    0018   1 !*                                                                   *
  19    0019   1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
  20    0020   1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
  21    0021   1 !*  CORPORATION.                                                      *
  22    0022   1 !*                                                                   *
  23    0023   1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
  24    0024   1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.           *
  25    0025   1 !*                                                                   *
  26    0026   1 !*                                                                   *
  27    0027   1 !*********************************************************************
  28    0028   1 !
  29    0029   1 !
  30    0030   1 !++
  31    0031   1 ! FACILITY:  FORTRAN support library
  32    0032   1 !
  33    0033   1 ! ABSTRACT:
  34    0034   1 !
  35    0035   1 !     This module contains the error handlers needed by
  36    0036   1 !     the common OTS for handling FORTRAN errors.  In particular
  37    0037   1 !     there is a handler for errors in OPEN/CLOSE where ERR=
  38    0038   1 !     means error return to caller rather than a transfer.
  39    0039   1 !     A second handler (FOR$$ERR_END_HND is provided
  40    0040   1 !     for I/O statements where the optional ERR= and END=
  41    0041   1 !     constructs require a transfer of control to the
  42    0042   1 !     user program rather than an an error return.
  43    0043   1 !     A third handler, FOR$$IOSTAT_HND is for auxilliary I/O statements
  44    0044   1 !     which either undwind with R0 containing an IOSTAT value or
  45    0045   1 !     resignal.
  46    0046   1 !     An argument specifies the cleanup to be performed if UNWIND occurs.
  47    0047   1 !
  48    0048   1 ! ENVIRONMENT:  User mode, AST level or not or mixed.
  49    0049   1 !               Note: this module is both shared (with no entry vectors) and non-shared
  50    0050   1 !               if FORTRAN compatibility routines call.
  51    0051   1 !
  52    0052   1 ! AUTHOR: Thomas N. Hastings, CREATION DATE: 03-Jun-77
  53    0053   1 !
  54    0054   1 ! MODIFIED BY:
  55    0055   1 !
  56    0056   1 !     Thomas N. Hastings, 03-Jun-77: VERSION 01
  57    0057   1 !     Steven B. Lionel, VAX/VMS V2.0
```

```
   58    0058  1 ! [Previous edit history deleted.  SBL 30-Sep-1982]
   59    0059  1 ! 1-019 - Look at FAO_COUNT in signal list to see where USER_PC is.  SBL 10-NOV-1980
   60    0060  1 ! 1-020 - Reset RAB$L_UBF and RAB$W_USZ in CLEANUP_LUB.   JAW-08-Jun-1981
   61    0061  1 ! 1-021 - Change OTS$$ data structure references to FOR$$.  SBL 30-Sep-1982
   62    0062  1 ! 1-022 - Look at FAB$W_IFI instead of LUB$W_IFI in CLEANUP_LUB.  QAR #1229.
   63    0063  1 !            SBL 7-Mar-1984
   64    0064  1 !--
   65    0065  1
```

```
  67    0066   1 !
  68    0067   1 !   PROLOGUE FILE:
  69    0068   1 !
  70    0069   1
  71    0070   1   REQUIRE 'RTLIN:FORPROLOG';                     ! FORTRAN definitions
  72    0136   1
  73    0137   1 !
  74    0138   1 !   TABLE OF CONTENTS:
  75    0139   1 !
  76    0140   1
  77    0141   1   FORWARD ROUTINE
  78    0142   1       FOR$$ERR_OPECLO,                           ! Error handler for OPEN/CLOSE
  79    0143   1       FOR$$ERR_ENDHND,                           ! ERR=/END= handler for I/O statements
  80    0144   1       FOR$$IOSTAT_HND,                           ! IOSTAT only handler
  81    0145   1       FOR$$IO_IN_PROG,                           ! I/O in progress handler
  82    0146   1       CLEANUP_LUB : NOVALUE;                     ! Perform appropriate LUB cleanup if UNWIND.
  83    0147   1                                                  ! signal list.
  84    0148   1
  85    0149   1 !
  86    0150   1 !   EQUATED SYMBOLS:
  87    0151   1 !
  88    0152   1 !       NONE
  89    0153   1 !
  90    0154   1 !   OWN STORAGE:
  91    0155   1 !
  92    0156   1 !       NONE
  93    0157   1 !
  94    0158   1 !   EXTERNAL REFERENCES:
  95    0159   1 !
  96    0160   1 !+
  97    0161   1 !   MAINTENANCE NOTE:  Since this module is called by FORTRAN compatibility
  98    0162   1 !   routines which are un-shared and the entry points are not vectored,
  99    0163   1 !   a separate copy of this module is linked with the user program when
 100    0164   1 !   the user calls a FORTRAN compatibility routine.  In order to prevent
 101    0165   1 !   data truncation errors from the linker, all external references are
 102    0166   1 !   of addressing mode general (rather than word displacement) even for
 103    0167   1 !   the same PSECT.
 104    0168   1 !-
 105    0169   1
 106    0170   1   EXTERNAL ROUTINE
 107    0171   1       FOR$$CB_GET : JSB_CB_GET NOVALUE,          ! Get current LUB/ISB/RAB
 108    0172   1                                                  ! Note: this non-shared routine is loaded if
 109    0173   1                                                  ! compatibility routines call, so can't reference
 110    0174   1                                                  ! FOR$$A_CUR_LUB directly.
 111    0175   1       FOR$$CB_POP : JSB_CB_POP NOVALUE,          ! Pop current LUB/ISB/RAB
 112    0176   1                                                  ! as specified by CCB.
 113    0177   1       FOR$$FP_MATCH : CALL_CCB NOVALUE,          ! Match FP in ISB chain
 114    0178   1       FOR$$FREE_VM,                              ! Free virtual memory
 115    0179   1       FOR$$CLOSE_FILE,                           ! RMS Close a file
 116    0180   1       FOR$$SIG_FATINT : NOVALUE,                 ! SIGNAL_STOP OTS$_FATINTERR
 117    0181   1       FOR$$SIG_DATCOR : NOVALUE,                 ! SIGNAL_STOP OTS$_INTDATCOR
 118    0182   1                                                  ! (FATAL INTERNAL ERROR IN RUN-TIME LIBRARY)
 119    0183   1       LIB$SIG_TO_RET;                            ! convert a SIGNAL to error return
 120    0184   1                                                  ! to caller of establisher with R0 set to signal value.
 121    0185   1
```

```
  123        0186   1   GLOBAL ROUTINE FOR$$ERR_OPECLO (                     ! Error condition handler for OPEN/CLOSE
  124        0187   1           SIG_ARGS_ADR,                               ! Adr. of SIGNAL args
  125        0188   1           MCH_ARGS_ADR,                               ! Adr. of mechanism args
  126        0189   1           ENB_ARGS_ADR)                               ! Adr.  of ENABLE declared args
  127        0190   1       =                                              ! Condition handlers always have values
  128        0191   1
  129        0192   1   !++
  130        0193   1   ! FUNCTIONAL DESCRIPTION:
  131        0194   1   !
  132        0195   1   !       FOR$$ERR_OPECLO is an error conditon handler established by
  133        0196   1   !       the OPEN and CLOSE statement procedures.  If the user specified
  134        0197   1   !       an ERR= keyword parameter, the handler unwinds the stack after
  135        0198   1   !       storing the signaled error condition in the saved image of R0.
  136        0199   1   !       Otherwise, FOR$$ERR_OPECLO just resignals by simply returning
  137        0200   1   !       SS$_RESIGNAL (to CHF).
  138        0201   1   !       If and when an UNWIND occurs, the ENABLE arg UNWIND_ACT_ADR
  139        0202   1   !       specifies whether the LUB/ISB/RAB is to be pop, returned, or no-opped.
  140        0203   1   !       It is not popped if it had not yet been pushed as indicated
  141        0204   1   !       by the ENABLE arg UNWIND_ACT_ADR.
  142        0205   1   !
  143        0206   1   !       If ERR= and IOSTAT were both specified, then the returned
  144        0207   1   !       value is the FORTRAN small integer error code.
  145        0208   1   !
  146        0209   1   ! FORMAL PARAMETERS:
  147        0210   1   !
  148        0211   1   !       SIG-ARG-ADR
  149        0212   1   !       SIG_ARGS_ADR.rl.ra        Adr. of Signal arg list
  150        0213   1   !       MCH_ARGS_ADR.rl.ra        Adr. of mechanism arg list
  151        0214   1   !       ENB_ARGS_ADR.rl.ra        Adr. of ENABLE arg list which contains:
  152        0215   1   !           ENABLE_COUNT.rbu.v  No. of longword following in ENABLE arg list
  153        0216   1   !           UNWIND_ACT_ADR.rl.r Adr. of longword containing UNWIND action code.
  154        0217   1   !                                 Any of FOR$K_UNWINDNOP, FOR$K_UNWINDPOP,
  155        0218   1   !                                 FOR$K_UNWINDRET.
  156        0219   1   !           [OPECLO_ADR.rlu.ra] Optional adr. of cannonical array of OPEN or CLOSE keyword
  157        0220   1   !                                 parameters after the encoded user parameter
  158        0221   1   !                                 list has been scanned and expanded into it.
  159        0222   1   !                                 Symbolic offsets into ENB_ARGS_ADR[1,OPEN$K_name] are of the
  160        0223   1   !                                 form OPEN$K_name as defined in FOROPN REQUIRE file.
  161        0224   1   !                                 If ommitted, assume no ERR= (DEFINE FILE, REWIND, etc)
  162        0225   1   !
  163        0226   1   ! IMPLICIT INPUTS:
  164        0227   1   !
  165        0228   1   !       FOR$$A_CUR_LUB            Adr. of current LUB/ISB/RAB or 0
  166        0229   1   !                                Note: obtained by calling FOR$$CB_GET
  167        0230   1   !                                      rather than directly.
  168        0231   1   !
  169        0232   1   ! IMPLICIT OUTPUTS:
  170        0233   1   !
  171        0234   1   !       SIG_ARGS_ADR[SIG$_USER_PC]    Set to user call PC to RTL
  172        0235   1   !
  173        0236   1   ! COMPLETION CODES:
  174        0237   1   !
  175        0238   1   !       SS$_RESIGNAL if no ERR= was specified
  176        0239   1   !       SS$_NORMAL if ERR= was specified (ignored by CHF on UNWIND)
  177        0240   1   !
  178        0241   1   ! SIDE EFFECTS:
  179        0242   1   !
```

```
  180    0243  1 |        If the user has specified ERR=, the stack is unwound to the
  181    0244  1 |        caller of the establisher (i.e., the user program) with the save image
  182    0245  1 |        of R0 set to the error status.
  183    0246  1 |        If no ERR= was specified,the error conditon is resignaled.
  184    0247  1 |        If UNWIND call, the current LUB/ISB/RAB may be popped or returned.
  185    0248  1 |--
  186    0249  1
  187    0250  2      BEGIN
  188    0251  2
  189    0252  2      BUILTIN
  190    0253  2          CALLG,
  191    0254  2          AP;
  192    0255  2
  193    0256  2      LITERAL                                          ! Define ENABLE arglist offsets
  194    0257  2          ENABLE_COUNT = 0,                            ! Offset in ENB_ARGS_ADR of no. of enable args following
  195    0258  2          UNWIND_ACT_ADR = 1,                          ! Adr. of longword containing
  196    0259  2                                                      ! UNWIND action code.
  197    0260  2          OPECLO_ADR = 2;                              ! Adr. of OPEN/CLOSE cannonical array
  198    0261  2
  199    0262  2      MAP
  200    0263  2          SIG_ARGS_ADR : REF BLOCK [, BYTE],          ! SIGNAL args
  201    0264  2          MCH_ARGS_ADR : REF BLOCK [, BYTE],          ! mechanism args
  202    0265  2          ENB_ARGS_ADR : REF VECTOR [OPECLO_ADR, LONG];  !ENABLE args list array
  203    0266  2
  204    0267  2      LOCAL
  205    0268  2          EST_FP : REF BLOCK [, BYTE],                ! Establisher's FP
  206    0269  2          SIG_PC_LOC: REF VECTOR [, LONG],            ! Location of user PC in signal list
  207    0270  2          OPECLO_ARRAY : REF VECTOR [OPEN$K_KEY_MAX + 1, LONG];   ! OPEN/CLOSE cannonical array
  208    0271  2
  209    0272  2      !+
  210    0273  2      ! If this is unwind condition, perform cleanup. since
  211    0274  2      ! Perform LUB cleanup indicated by EBABLE arg UNWIND_ACT_ADR
  212    0275  2      ! (set by the establisher).
  213    0276  2      !-
  214    0277  2
  215    0278  3      IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_COND_ID;, BYTE] EQL (SS$_UNWIND^-3)
  216    0279  2      THEN
  217    0280  3          BEGIN
  218    0281  3          CLEANUP_LUB (..ENB_ARGS_ADR [UNWIND_ACT_ADR]);
  219    0282  3          RETURN SS$_NORMAL;
  220    0283  2          END;
  221    0284  2
  222    0285  2      OPECLO_ARRAY = .ENB_ARGS_ADR [OPECLO_ADR];
  223    0286  2
  224    0287  2      !+
  225    0288  2      ! If this is not a FOR$ error or if another RTL handler has seen this
  226    0289  2      ! error (noted by signal argument for user PC being non-zero) then
  227    0290  2      ! just resignal.
  228    0291  2      !-
  229    0292  2
  230    0293  2      IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_FAC_NO;, BYTE] NEQ FOR$K_FAC_NO
  231    0294  2      THEN
  232    0295  2          RETURN SS$_RESIGNAL;
  233    0296  2
  234    0297  2      SIG_PC_LOC = SIG_ARGS_ADR [CHF$L_SIG_ARG1] + (.SIG_ARGS_ADR [CHF$L_SIG_ARG1] * %UPVAL);
  235    0298  2      IF .SIG_PC_LOC [0] NEQ 0
  236    0299  2      THEN
```

```
237    0300  2              RETURN SS$_RESIGNAL;
238    0301
239    0302  2      !+
240    0303  2      ! Check if user provided ERR= keyword or not.  If yes, convert signal to
241    0304  2      ! a return to the caller of the establisher with condition value in R0.
242    0305  2      ! If IOSTAT is present, act as if ERR= is also.
243    0306  2      ! If caller omitted OPECLO_ADR entry in ENB_ARGS_ADR, treat as if no ERR=.
244    0307         !-
245    0308
246    0309  2      IF .ENB_ARGS_ADR [ENABLE_COUNT] GEQU OPECLO_ADR AND (.OPECLO_ARRAY [OPEN$K_ERR] OR .OPECLO_ARRAY [
247    0310  2          OPEN$K_IOSTAT]) NEQ 0
248    0311  2      THEN
249    0312  2          BEGIN
250    0313  3
251    0314  3          !+
252    0315  3          ! If IOSTAT was specified, store the value.
253    0316  3          !-
254    0317  3
255    0318  3          IF .OPECLO_ARRAY [OPEN$K_IOSTAT] NEQ 0
256    0319  3          THEN
257    0320  4              BEGIN
258    0321  4
259    0322  4              LOCAL
260    0323  4                  IOSTAT;
261    0324  4
262    0325  4              IOSTAT = .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_CODE;, BYTE];
263    0326  4
264    0327  4              IF .OPECLO_ARRAY [OPEN$K_IOSTAT_L]
265    0328  4              THEN
266    0329  4                  .OPECLO_ARRAY [OPEN$K_IOSTAT] = .IOSTAT
267    0330  4              ELSE
268    0331  5                  BEGIN
269    0332  5
270    0333  5                  LOCAL
271    0334  5                      IOSTAT_ADR : REF BLOCK [, BYTE];
272    0335  5
273    0336  5                  IOSTAT_ADR = .OPECLO_ARRAY [OPEN$K_IOSTAT];
274    0337  5                  IOSTAT_ADR [0, 0, 16, 0] = .IOSTAT;
275    0338  4                  END;
276    0339  4
277    0340  3              END;
278    0341  3
279    0342  3          IF NOT CALLG (.AP, LIB$SIG_TO_RET) THEN FOR$$SIG_FATINT ()
280    0343  3
281    0344  3          END
282    0345  2      ELSE
283    0346  2
284    0347  2      !+
285    0348  2      ! No ERR=, so set user call PC saved in stack frame of establisher and RESIGNAL
286    0349  2      !-
287    0350  2
288    0351  3          BEGIN
289    0352  3          EST_FP = .MCH_ARGS_ADR [CHF$L_MCH_FRAME];
290    0353  3          SIG_PC_LOC [0] = .EST_FP [SF$L_SAVE_PC];
291    0354  3
292    0355  2          END;                                          ! End no ERR=
293    0356  2
```

```
:   294        0357  2        !+
:   295        0358  2        ! Return resignal condition (ignored if SYS$UNWIND called).
:   296        0359  2        !-
:   297        0360  2
:   298        0361  2        RETURN SS$_RESIGNAL
:   299        0362  1        END;                                    ! End of FOR$$ERR_OPECLO handler

                             .TITLE   FOR$$ERROR Internal FORTRAN error handling modu
:                                              le
                             .IDENT   \1-022\

                             .EXTRN   FOR$$CB_GET, FOR$$CB_POP
                             .EXTRN   FOR$$FP_MATCH, FOR$$FREE_VM
                             .EXTRN   FOR$$CLOSE_FILE
                             .EXTRN   FOR$$SIG_FATINT
                             .EXTRN   FOR$$SIG_DATCOR
                             .EXTRN   LIB$SIG_TO_RET

                             .PSECT   _FOR$CODE,NOWRT,  SHR,  PIC,2

                       000C 00000      .ENTRY   FOR$$ERR_OPECLO, Save R2,R3          : 0186
                        52    04 AC D0 00002      MOVL     SIG_ARGS_ADR, R2            : 0278
00000124  8F    04 A2  19    03 ED 00006      CMPZV    #3, #25, 4(R2), #292
                        10    12 00010      BNEQ     1$
                        50    0C AC D0 00012      MOVL     ENB_ARGS_ADR, R0           : 0281
                        04    B0 DD 00016      PUSHL    @4(R0)
               0000V CF 01    FB 00019      CALLS    #1, CLEANUP_LUB
                        50    01 D0 0001E      MOVL     #1, R0                        : 0282
                        04    00021      RET
                        53    0C AC D0 00022 1$: MOVL    ENB_ARGS_ADR, R3            : 0285
                        50    08 A3 D0 00026      MOVL     8(R3), OPECLO_ARRAY
               18  06 A2 0C    00 ED 0002A      CMPZV    #0, #12, 6(R2), #24         : 0293
                        54    12 00030      BNEQ     5$
                        51    08 A2 D0 00032      MOVL     8(R2), R1                  : 0297
                        51    08 A241 DE 00036      MOVAL    8(R2)[R1], SIG_PC_LOC
                        61    D5 0003B      TSTL     (SIG_PC_LOC)                     : 0298
                        47    12 0003D      BNEQ     5$
                        02    63 D1 0003F      CMPL     (R3), #2                      : 0309
                        36    1F 00042      BLSSU    4$
               53  0C A0 58 A0 C9 00044      BISL3    88(OPECLO_ARRAY), 12(OPECLO_ARRAY), R3
                        2E    13 0004A      BEQL     4$
                        53    58 A0 D0 0004C      MOVL     88(OPECLO_ARRAY), R3       : 0310
                        15    13 00050      BEQL     3$                              : 0318
               52  04 A2 0C    03 EF 00052      EXTZV    #3, #12, 4(R2), IOSTAT      : 0325
                        05    64 A0 E9 00058      BLBC     100(OPECLO_ARRAY), 2$      : 0327
                        63    52 D0 0005C      MOVL     IOSTAT, (R3)                 : 0329
                        06    11 0005F      BRB      3$
                        50    53 D0 00061 2$: MOVL     R3, IOSTAT_ADR               : 0336
                        60    52 B0 00064      MOVW     IOSTAT, (IOSTAT_ADR)        : 0337
               00000000G 00    6C FA 00067 3$: CALLG    (AP), LIB$SIG_TO_RET        : 0342
                        15    50 E8 0006E      BLBS     R0, 5$
               00000000G 00    00 FB 00071      CALLS    #0, FOR$$SIG_FATINT
                        0C    11 00078      BRB      5$
                        50    08 AC D0 0007A 4$: MOVL    MCH_ARGS_ADR, R0           : 0352
                        50    04 A0 D0 0007E      MOVL     4(R0), EST_FP
                        61    10 A0 D0 00082      MOVL     16(EST_FP), (SIG_PC_LOC)   : 0353
```

```
                                    50     0918  8F  3C 00086 5$:    MOVZWL  #2328, R0              ; 0361
                                              04 0008B         RET                                 ; 0362
```

; Routine Size: 140 bytes,   Routine Base: _FOR$CODE + 0000

; 300   0363 1

```
302     0364  1  GLOBAL ROUTINE FOR$$ERR_ENDHND (                    ! FORTRAN I/O statement ERR=/END= error condition handler
303     0365  1          SIG_ARGS_ADR,                               ! Adr. of signal arg list
304     0366  1          MCH_ARGS_ADR,                               ! Adr. of mechanism arg list
305     0367  1          ENB_ARGS_ADR)                               ! Adr. of ENABLE arg list
306     0368  1      =                                               ! Return status for a condition handler
307     0369  1
308     0370  1  !++
309     0371  1  ! FUNCTIONAL DESCRIPTION:
310     0372  1  !
311     0373  1  !       FOR$$ERR_ENDHND is an error condition handler established
312     0374  1  !       by each I/O statement which has an  ERR= and END= error transfer
313     0375  1  !       mechanism (as an option of the user program).
314     0376  1  !
315     0377  1  !       If the signaled condition is FOR$_ENDDURREA (24="END-OF_FILE DURING READ')
316     0378  1  !       and an END= has been specified by the user in his I/O statement
317     0379  1  !       (.END_EQL_ADR NEQ 0), the handler unwinds to the user specified address (by calling
318     0380  1  !       SYS$UNWIND with depth equal to CHF$L_MCH_DEPTH + ..INCR_DEPTH_ADR + 1)
319     0381  1  !       and new_PC equal to ..END_EQL_ADR.
320     0382  1  !       Otherwise, if an ERR= had been specified by the user in his I/O statement
321     0383  1  !       (ERR_EQUL NEQ 0), the handler unwinds to the user specified address
322     0384  1  !       by calling SYS$UNWIND with depth equal to CHF$L_MCH_DEPTH + ..INCR_DEPTH_ADR + 1
323     0385  1  !       and new_PC equal to ..ERR_EQL_ADR.
324     0386  1  !
325     0387  1  !       If neither of the above cases holds, the error is resignaled
326     0388  1  !       so that a user handler or the OTS default handler will get invoked.
327     0389  1  !       If UNWIND occurs, the appropriate cleanup takes place,
328     0390  1  !       as indicated by the establisher in the ENABLE arg UNWIND_ACT_ADR.
329     0391  1  !       If FOR$K_UNWINDPOP is indicated, the current LUB/ISB/RAB is popped.
330     0392  1  !       If FOR$K_UNWINDRET is indicated, the LUB/ISB/RAB is returned and the
331     0393  1  !       file closed.
332     0394  1  !       Otherwise (FOR$K_UNWINDNOP) nothing is done.
333     0395  1  !
334     0396  1  ! FORMAL PARAMETERS:
335     0397  1  !
336     0398  1  !       SIG_ARGS_ADR.ml.ra        Adr. of signal arg list
337     0399  1  !       MCH_ARGS_ADR.ml.ra        Adr. of mechanism arg list
338     0400  1  !       ENB_ARGS_ADR.ml.ra        Adr. of ENABLE arg list which contains:
339     0401  1  !          UNWIND_ACT_ADR.rl.r    Adr. of longword contining UNWIND action code.
340     0402  1  !                                 Any of FOR$K_UNWINDNOP, FOR$K_UNWINDPOP,
341     0403  1  !                                 FOR$K_UNWINDRET.
342     0404  1  !          ERR_EQL_ADR.ra.r       Adr. of longword containing Adr. of the user address
343     0405  1  !                                 to be transferred to or 0 on any error condition
344     0406  1  !          END_EQL_ADR.ra.r       Adr of longword containing Adr. of the user address
345     0407  1  !                                 to be transferred to or 0 on end-of-file
346     0408  1  !          INCR_DEPTH_ADR.rl.r    Adr. of longword containing Incremental no. of frames between the establishe
347     0409  1  !                                 and the users program (usually 0 or 1).
348     0410  1  ! Note: All parameters to a condition handler must be addresses of values in BLISS if used in an ENABLE.
349     0411  1  !
350     0412  1  ! IMPLICIT INPUTS:
351     0413  1  !
352     0414  1  !       FOR$$A_CUR_LUB            Adr. of current LUB/ISB/RAB or 0
353     0415  1  !                                Note: obtained by calling FOR$$CB_GET rather than directly.
354     0416  1  !
355     0417  1  ! IMPLICIT OUTPUTS:
356     0418  1  !
357     0419  1  !       SIG_ARGS_ADR[SIG$_USER_PC]     Set to user call PC to RTL
358     0420  1  !
```

```
359    0421  1  ! COMPLETION CODES:
360    0422  1  !
361    0423  1  !     SS$_RESIGNAL if no ERR= or END= was specified by user, so that
362    0424  1  !     a user handler or the default OTS handler will get a chance.
363    0425  1  !     SS$_NORMAL if unwind called (although ignored if unwind called)
364    0426  1  !
365    0427  1  ! SIDE EFFECTS:
366    0428  1  !
367    0429  1  !     If END= and EOF OR ERR= was specified, the stack is unwound
368    0430  1  !     to user and new_PC is set from ..END_EQL_ADR or .ERR_EQL_ADR.
369    0431  1  !     If unwind, the current LUB/ISB/RAB may be popped or returned.
370    0432  1  !--
371    0433  1
372    0434  2     BEGIN
373    0435  2
374    0436  2     LOCAL
375    0437  2         EST_FP : REF BLOCK [, BYTE],              ! Establisher's FP
376    0438  2         SIG_PC_LOC: REF VECTOR [, LONG];          ! Location of user PC in signal list
377    0439  2
378    0440  2     LITERAL                                       ! Declare offsets in ENABLE VECTOR arg list
379    0441  2         UNWIND_ACT_ADR = 1,                       ! UNWIND action code
380    0442  2         ERR_EQL_ADR = 2,                         ! ERR= adr or 0
381    0443  2         END_EQL_ADR = 3,                         ! END= adr or 0
382    0444  2         INCR_DEPTH_ADR = 4;                      ! incremental depth
383    0445  2
384    0446  2     MAP
385    0447  2         SIG_ARGS_ADR : REF BLOCK [, BYTE],        ! SIGNAL arg list
386    0448  2         MCH_ARGS_ADR : REF BLOCK [, BYTE],        ! mechanism arg list
387    0449  2         ENB_ARGS_ADR : REF VECTOR [INCR_DEPTH_ADR + 1, LONG];   ! ENABLE arg list
388    0450  2
389    0451  2
390    0452  2     !+
391    0453  2     ! Check for unwinding since handler gets called when it does an unwind.
392    0454  2     ! If unwind, perform cleanup indicated by ENABLE arg UNWIND_ACT_ADR.
393    0455  2     ! Then return to the unwinder to keep unwinding (return value ignored).
394    0456  2     !-
395    0457  2
396    0458  2     IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_COND_ID;, BYTE] EQL (SS$_UNWIND^-3)
397    0459  2     THEN
398    0460  3         BEGIN
399    0461  3         CLEANUP_LUB (..ENB_ARGS_ADR [UNWIND_ACT_ADR]);
400    0462  3         RETURN SS$_NORMAL;
401    0463  2         END;
402    0464  2
403    0465  2     !+
404    0466  2     ! If error is not a FOR$ error or if another RTL handler has seen
405    0467  2     ! this error then resignal.
406    0468  2     !-
407    0469  2
408    0470  2     IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_FAC_NO;, BYTE] NEQ FOR$K_FAC_NO
409    0471  2     THEN
410    0472  2         RETURN SS$_RESIGNAL;
411    0473  2
412    0474  2     SIG_PC_LOC = SIG_ARGS_ADR [CHF$L_SIG_ARG1] + (.SIG_ARGS_ADR [CHF$L_SIG_ARG1] * %UPVAL);
413    0475  2     IF .SIG_PC_LOC [0] NEQ 0
414    0476  2     THEN
415    0477  2         RETURN SS$_RESIGNAL;
```

FOR$$ERROR   Internal FORTRAN error handling module   I 5
1-022   16-Sep-1984 00:20:31   VAX-11 Bliss-32 V4.0-742   Page 11
                                                                    14-Sep-1984 12:31:54   [FORRTL.SRC]FORERROR.B32;1   (4)

**F

```
416     0478    2
417     0479    2           !+
418     0480    2           ! Check for END= and ERR=.
419     0481    2           ! If this is end-of-file (during read)
420     0482    2           ! Unwind to the user with the new_pc being .END_ADR and with
421     0483    2           ! R0 as an IOSTAT value of -1.
422     0484    2           !-
423     0485    2
424     0486    2           IF ..ENB_ARGS_ADR [END_EQL_ADR] NEQA 0 AND .SIG_ARGS_ADR [CHF$L_SIG_NAME] EQL FOR$_ENDDURREA
425     0487    2           THEN
426     0488    2               BEGIN
427     0489    3
428     0490    3               LOCAL
429     0491    3                   T;
430     0492    3
431     0493    3               MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = -1;
432     0494    3               T = .MCH_ARGS_ADR [CHF$L_MCH_DEPTH] + ..ENB_ARGS_ADR [INCR_DEPTH_ADR] + 1;
433     0495    3
434     0496    4               IF $UNWIND (DEPADR = T, NEWPC = ..ENB_ARGS_ADR [END_EQL_ADR])
435     0497    3               THEN
436     0498    3                   RETURN SS$_NORMAL
437     0499    3               ELSE
438     0500    3                   FOR$$SIG_FATINT ()
439     0501    3
440     0502    2               END;
441     0503    2
442     0504    2           !+
443     0505    2           ! If this is an error, and ERR= was specified by the user,
444     0506    2           ! Unwind to the user with the new_pc being .ERR_ADR and
445     0507    2           ! with R0 set to the proper IOSTAT value.
446     0508    2           !-
447     0509    2
448     0510    2           IF ..ENB_ARGS_ADR [ERR_EQL_ADR] NEQA 0
449     0511    2           THEN
450     0512    3               BEGIN
451     0513    3
452     0514    3               LOCAL
453     0515    3                   T;
454     0516    3
455     0517    3               IF .SIG_ARGS_ADR [CHF$L_SIG_NAME] EQL FOR$_ENDDURREA
456     0518    3               THEN
457     0519    3                   MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = -1
458     0520    3               ELSE
459     0521    3                   MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_CODE;, BYTE];
460     0522    3
461     0523    3               T = .MCH_ARGS_ADR [CHF$L_MCH_DEPTH] + ..ENB_ARGS_ADR [INCR_DEPTH_ADR] + 1;
462     0524    3
463     0525    4               IF $UNWIND (DEPADR = T, NEWPC = ..ENB_ARGS_ADR [ERR_EQL_ADR])
464     0526    3               THEN
465     0527    3                   RETURN SS$_NORMAL
466     0528    3               ELSE
467     0529    3                   FOR$$SIG_FATINT ()
468     0530    3
469     0531    2               END;
470     0532    2
471     0533    2           !+
472     0534    2           ! If neither END= nor ERR= specified by user.
```

```
: 473    0535  2      ! Scan back from frame of establisher to frame of routine to called by user.
: 474    0536  2      ! Set user CALL PC to library in SIGNAL arg list.
: 475    0537  2      ! Just indicate to the condition handling facility to resignal the condition
: 476    0538  2      !  so that a user supplied handler or the OTS default handler will get a chance to handle.
: 477    0539  2      !-
: 478    0540
: 479    0541  2      EST_FP = .MCH_ARGS_ADR [CHF$L_MCH_FRAME];
: 480    0542  2
: 481    0543  2      DECR I FROM ..ENB_ARGS_ADR [INCR_DEPTH_ADR] TO 1 DO
: 482    0544  2          EST_FP = .EST_FP [SF$L_SAVE_FP];
: 483    0545  2
: 484    0546  2      SIG_PC_LOC [0] = .EST_FP [SF$L_SAVE_PC];
: 485    0547  2
: 486    0548  2      RETURN SS$_RESIGNAL
: 487    0549  1      END;                                    !End of FOR$$ERR_ENDHND
```

```
                                                     .EXTRN   SYS$UNWIND

                                  007C 00000          .ENTRY   FOR$$ERR_ENDHND, Save R2,R3,R4,R5,R6    : 0364
                  56 00000000G  00  9E 00002          MOVAB    FOR$$SIG_FATINT, R6
                  55 00000000G  00  9E 00009          MOVAB    SYS$UNWIND, R5
                           5E  08  C2 00010           SUBL2    #8, SP
                           52  04  AC D0 00013         MOVL     SIG_ARGS_ADR, R2                        : 0458
                           53  04  A2 9E 00017         MOVAB    4(R2), R3
00000124  8F            63  19  03  ED 0001B           CMPZV    #3, #25, (R3), #292
                           0F  12 00024               BNEQ     1$
                           50  0C  AC D0 00026         MOVL     ENB_ARGS_ADR, R0                        : 0461
                           04  B0  DD 0002A            PUSHL    a4(R0)
                    0000V CF  01  FB 0002D             CALLS    #1, CLEANUP_LUB
                         0080  31 00032               BRW      7$                                       : 0462
         18       02  A3  0C  00  ED 00035 1$:         CMPZV    #0, #12, 2(R3), #24                     : 0470
                           0B  12 0003B               BNEQ     2$
                           50  08  A2 D0 0003D         MOVL     8(R2), R0                               : 0474
                           54  08 A240 DE 00041        MOVAL    8(R2)[R0], SIG_PC_LOC
                           64  D5 00046               TSTL     (SIG_PC_LOC)                             : 0475
                           03  13 00048 2$:           BEQL     3$
                         0089  31 0004A               BRW      12$
                           52  0C  AC D0 0004D 3$:     MOVL     ENB_ARGS_ADR, R2                        : 0486
                           0C  B2  D5 00051            TSTL     a12(R2)
                           2A  13 00054               BEQL     4$
                001880C4 8F  63  D1 00056             CMPL     (R3), #1605828
                           21  12 0005D               BNEQ     4$
                           50  08  AC D0 0005F         MOVL     MCH_ARGS_ADR, R0                        : 0493
                      0C  A0  01  CE 00063             MNEGL    #1, -12(R0)
                  50  08  A0  10  B2 C1 00067          ADDL3    a16(R2), 8(R0), R0                      : 0494
                           6E  01  A0 9E 0006D         MOVAB    1(R0), t
                           0C  B2  DD 00071            PUSHL    a12(R2)                                  : 0496
                           04  AE  9F 00074            PUSHAB   t
                           65  02  FB 00077            CALLS    #2, SYS$UNWIND
                           38  50  E8 0007A            BLBS     R0, 7$
                           66  00  FB 0007D            CALLS    #0, FOR$$SIG_FATINT                     : 0500
                           08  B2  D5 00080 4$:        TSTL     a8(R2)                                  : 0510
                           37  13 00083               BEQL     9$
                           50  08  AC D0 00085         MOVL     MCH_ARGS_ADR, R0                        : 0519
                001880C4 8F  63  D1 00089             CMPL     (R3), #1605828                           : 0517
```

```
                                    06 12 00090            BNEQ    5$                              0519
                          OC  AO    01 CE 00092            MNEGL   #1, 12(RO)
                                    06 11 00096            BRB     6$
      OC  AO          63        OC  03 EF 00098 5$:        EXTZV   #3, #12, (R3), 12(RO)          0521
                      50    08  AO  10 B2 C1 0009E 6$:      ADDL3   @16(R2), 8(RO), RO             0523
                           04  AE  01 AO 9E 000A4          MOVAB   1(RO), T
                               08  B2 DD 000A9            PUSHL   @8(R2)                          0525
                               08  AE 9F 000AC            PUSHAB  T
                                    65 02 FB 000AF         CALLS   #2, SYS$UNWIND
                                    04 50 E9 000B2         BLBC    RO, 8$
                                    50 01 DO 000B5 7$:     MOVL    #1, RO                          0527
                                    04 000B8               RET
                                    66 00 FB 000B9 8$:     CALLS   #0, FOR$$SIG_FATINT             0529
                               50  08 AC DO 000BC 9$:      MOVL    MCH_ARGS_ADR, RO               0541
                               50  04 AO DO 000CO          MOVL    4(RO), EST_FP
                      51    10  B2  01 C1 000C4            ADDL3   #1, @16(R2), I                  0543
                                    04 11 000C9            BRB     11$
                               50  OC AO DO 000CB 10$:     MOVL    12(EST_FP), EST_FP             0544
                           F9  51 F5 000CF 11$:           SOBGTR  I, 10$
                           64  10 AO DO 000D2            MOVL    16(EST_FP), (SIG_PC_LOC)        0546
                      50  0918 8F 3C 000D6 12$:          MOVZWL  #2328, RO                        0548
                                    04 000DB               RET                                    0549
```

; Routine Size: 220 bytes,   Routine Base: _FOR$CODE + 008C

; 488   0550 1

FOR$$ERROR
1-022
Internal FORTRAN error handling module
L 5
16-Sep-1984 00:20:31   VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:31:54   [FORRTL.SRC]FORERROR.B32;1
Page 14
(5)

```
490   0551  1  GLOBAL ROUTINE FOR$$IOSTAT_HND (              ! FORTRAN I/O IOSTAT handler
491   0552  1           SIG_ARGS_ADR,                        ! Adr. of signal arg list
492   0553  1           MCH_ARGS_ADR,                        ! Adr. of mechanism arg list
493   0554  1           ENB_ARGS_ADR)                        ! Adr. of ENABLE arg list
494   0555  1      =                                         ! Return status for a condition handler
495   0556  1
496   0557  1  !++
497   0558  1  ! FUNCTIONAL DESCRIPTION:
498   0559  1  !
499   0560  1  !      FOR$$IOSTAT_HND is an error condition handler established by each
500   0561  1  !      auxilliary I/O statement which can have as optional arguments
501   0562  1  !      ERR= and IOSTAT=.
502   0563  1  !
503   0564  1  !      If the enable argument ERR_EQL_ADR is non zero, FOR$$IOSTAT_HND
504   0565  1  !      unwinds with the saved R0 set to the appropriate IOSTAT small
505   0566  1  !      integer FORTRAN error number.  If ERR_EQL_ADR is zero, then it
506   0567  1  !      is assumed that no ERR= is present and the error is resignalled.
507   0568  1  !      Note that the unwind is not done to the ERR= address, rather the
508   0569  1  !      compiled code makes a test of the returned value and branches
509   0570  1  !      to the designated ERR= statement itself.
510   0571  1  !
511   0572  1  !      If UNWIND occurs, the appropriate cleanup takes place,
512   0573  1  !      as indicated by the establisher in the ENABLE arg UNWIND_ACT_ADR.
513   0574  1  !      If FOR$K_UNWINDPOP is indicated, the current LUB/ISB/RAB is popped.
514   0575  1  !      If FOR$K_UNWINDRET is indicated, the LUB/ISB/RAB is returned and the
515   0576  1  !      file closed.
516   0577  1  !      Otherwise (FOR$K_UNWINDNOP) nothing is done.
517   0578  1  !
518   0579  1  ! FORMAL PARAMETERS:
519   0580  1  !
520   0581  1  !      SIG_ARGS_ADR.ml.ra      Adr. of signal arg list
521   0582  1  !      MCH_ARGS_ADR.ml.ra      Adr. of mechanism arg list
522   0583  1  !      ENB_ARGS_ADR.ml.ra      Adr. of ENABLE arg list which contains:
523   0584  1  !       UNWIND_ACT_ADR.rl.r Adr. of longword contining UNWIND action code.
524   0585  1  !                           Any of FOR$K_UNWINDNOP, FOR$K_UNWINDPOP,
525   0586  1  !                           FOR$K_UNWINDRET.
526   0587  1  !       ERR_EQL_ADR.rl.v    0 if there is no ERR= on the statement
527   0588  1  !                           1 if there is an ERR= present.
528   0589  1  ! Note: All parameters to a condition handler must be addresses of values in BLISS if used in an ENABLE.
529   0590  1  !
530   0591  1  ! IMPLICIT INPUTS:
531   0592  1  !
532   0593  1  !      FOR$$A_CUR_LUB          Adr. of current LUB/ISB/RAB or 0
533   0594  1  !                              Note: obtained by calling FOR$$CB_GET rather than directly.
534   0595  1  !
535   0596  1  ! IMPLICIT OUTPUTS:
536   0597  1  !
537   0598  1  !      MCH_ARGS_ADR [CHF$L_MCH_SAVR0]  Set to an IOSTAT value
538   0599  1  !
539   0600  1  ! COMPLETION CODES:
540   0601  1  !
541   0602  1  !      SS$_RESIGNAL if no ERR= or END= was specified by user, so that
542   0603  1  !      a user handler or the default OTS handler will get a chance.
543   0604  1  !      SS$_NORMAL if unwind called (although ignored if unwind called)
544   0605  1  !
545   0606  1  ! SIDE EFFECTS:
546   0607  1  !
```

M 5

```
547    0608   1 !        If ERR= was specified, the stack is unwound to the user.
548    0609   1 !        If unwind, the current LUB/ISB/RAB may be popped or returned.
549    0610   1 !--
550    0611   1
551    0612   2        BEGIN
552    0613   2
553    0614   2        LOCAL
554    0615   2            EST_FP : REF BLOCK [, BYTE],                    ! Establisher's FP
555    0616   2            SIG_PC_LOC: REF VECTOR [, LONG];                ! Location of user PC in signal list
556    0617
557    0618   2        LITERAL                                            ! Declare offsets in ENABLE VECTOR arg list
558    0619   2            UNWIND_ACT_ADR = 1,                            ! UNWIND action code
559    0620   2            ERR_EQL_ADR = 2;                               ! ERR= present, 1 or 0
560    0621   2
561    0622   2        MAP
562    0623   2            SIG_ARGS_ADR : REF BLOCK [, BYTE],             ! SIGNAL arg list
563    0624   2            MCH_ARGS_ADR : REF BLOCK [, BYTE],             ! mechanism arg list
564    0625   2            ENB_ARGS_ADR : REF VECTOR [ERR_EQL_ADR + 1, LONG];    ! ENABLE arg list
565    0626   2
566    0627   2        !+
567    0628   2        ! Check for unwinding since handler gets called when it does an unwind.
568    0629   2        ! If unwind, perform cleanup indicated by ENABLE arg UNWIND_ACT_ADR.
569    0630   2        ! Then return to the unwinder to keep unwinding (return value ignored).
570    0631   2        !-
571    0632   2
572    0633   2        IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_COND_ID;, BYTE] EQL (SS$_UNWIND^-3)
573    0634   2        THEN
574    0635   2            BEGIN
575    0636   3            CLEANUP_LUB (..ENB_ARGS_ADR [UNWIND_ACT_ADR]);
576    0637   3            RETURN SS$_NORMAL;
577    0638   3            END;
578    0639   2
579    0640   2        !+
580    0641   2        ! If this is not a FOR$ error or if another RTL handler has seen this
581    0642   2        ! error (noted by signal argument for user PC being non-zero) then
582    0643   2        ! just resignal.
583    0644   2        !-
584    0645
585    0646   2        IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_FAC_NO;, BYTE] NEQ FOR$K_FAC_NO
586    0647   2        THEN
587    0648   2            RETURN SS$_RESIGNAL;
588    0649   2        SIG_PC_LOC = SIG_ARGS_ADR [CHF$L_SIG_ARG1] + (.SIG_ARGS_ADR [CHF$L_SIG_ARG1] * %UPVAL);
589    0650   2        IF .SIG_PC_LOC [0] NEQ 0
590    0651   2        THEN
591    0652   2            RETURN SS$_RESIGNAL;
592    0653   2
593    0654   2        !+
594    0655   2        ! If this is an error, and ERR= was specified by the user,
595    0656   2        ! Unwind to the user with saved R0 being the IOSTAT value.
596    0657   2        !-
597    0658   2
598    0659   2        IF ..ENB_ARGS_ADR [ERR_EQL_ADR] NEQA 0
599    0660   2        THEN
600    0661   3            BEGIN
601    0662   3            MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_CODE;, BYTE];
602    0663   3
603    0664   4            IF $UNWIND ()
```

```
;    604        0665  3                          THEN
;    605        0666  3                              RETURN SS$_NORMAL
;    606        0667  3                          ELSE
;    607        0668  3                              FOR$$SIG_FATINT ();
;    608        0669  3
;    609        0670  2                          END;
;    610        0671  2
;    611        0672  2              !+
;    612        0673  2              ! If ERR= not specified by the user
;    613        0674  2              ! scan back from frame of establisher to frame of routine to called by user.
;    614        0675  2              ! Set user CALL PC to library in SIGNAL arg list.
;    615        0676  2              ! Just indicate to the condition handling facility to resignal the condition
;    616        0677  2              ! so that a user supplied handler or the OTS default handler will get a chance to handle.
;    617        0678  2              !-
;    618        0679  2
;    619        0680  2              EST_FP = .MCH_ARGS_ADR [CHF$L_MCH_FRAME];
;    620        0681  2              SIG_PC_LOC [0] = .EST_FP [SF$C_SAVE_PC];
;    621        0682  2              RETURN SS$_RESIGNAL
;    622        0683  1              END;                                              !End of FOR$$IOSTAT_HND
```

```
                                   000C 00000            .ENTRY  FOR$$IOSTAT_HND, Save R2,R3          : 0551
                            52     04 AC D0 00002         MOVL    SIG_ARGS_ADR, R2                     : 0633
00000124  8F    04 A2      19     03 ED 00006         CMPZV   #3, #25, 4(R2), #292
                           0E     12 00010             BNEQ    1$
                           50     0C AC D0 00012         MOVL    ENB_ARGS_ADR, R0                     : 0636
                           04     B0 DD 00016         PUSHL   a4(R0)
                 0000V CF  01     FB 00019             CALLS   #1, CLEANUP_LUB
                           35     11 0001E             BRB     2$                                      : 0637
       18     06 A2        0C     00 ED 00020 1$:     CMPZV   #0, #12, 6(R2), #24                      : 0646
                           44     12 00026             BNEQ    5$
                           50     08 A2 D0 00028         MOVL    8(R2), R0                             : 0649
                           53     08 A240 DE 0002C     MOVAL   8(R2)[R0], SIG_PC_LOC
                           63     D5 00031             TSTL    (SIG_PC_LOC)                            : 0650
                           37     12 00033             BNEQ    5$
                           50     0C AC D0 00035         MOVL    ENB_ARGS_ADR, R0                     : 0659
                           08     B0 D5 00039             TSTL    a8(R0)
                           22     13 0003C             BEQL    4$
                           50     08 AC D0 0003E         MOVL    MCH_ARGS_ADR, R0                     : 0662
   0C  A0     04 A2        0C     03 EF 00042         EXTZV   #3, #12, 4(R2), 12(R0)
                           7E     7C 00049             CLRQ    -(SP)
               00000000G   00     02 FB 0004B         CALLS   #2, SYS$UNWIND                          : 0664
                           04     50 E9 00052             BLBC    R0, 3$
                           50     01 D0 00055 2$:     MOVL    #1, R0                                   : 0666
                           04     00058             RET
               00000000G   00     00 FB 00059 3$:     CALLS   #0, FOR$$SIG_FATINT                     : 0668
                           50     08 AC D0 00060 4$:     MOVL    MCH_ARGS_ADR, R0                     : 0680
                           50     04 A0 D0 00064         MOVL    4(R0), EST_FP
                           63     10 A0 D0 00068         MOVL    16(EST_FP), (SIG_PC_LOC)              : 0681
                           50     0918 8F 3C 0006C 5$:  MOVZWL  #2328, R0                             : 0682
                           04     00071             RET                                                : 0683
```

; Routine Size: 114 bytes,   Routine Base: _FOR$CODE + 0168

; 623   0684 1

```
 625   0685  1  GLOBAL ROUTINE FOR$$IO_IN_PROG (                      ! I/O in progress handler
 626   0686  1          SIG_ARGS_ADR,                                 ! Address of signal arg list
 627   0687  1          MCH_ARGS_ADR)                                 ! Address of mechanism arg list
 628   0688  1      =
 629   0689  1
 630   0690  1  !++
 631   0691  1  ! FUNCTIONAL DESCRIPTION:
 632   0692  1  !
 633   0693  1  !       FOR$$IO_IN_PROG is a special handler that is designed to
 634   0694  1  !       allow the Run-Time Library to clean I/O that is in progress
 635   0695  1  !       when an error occurs during the processing of a multi-call
 636   0696  1  !       I/O statement.  For example, if evaluation of a variable
 637   0697  1  !       list item in a WRITE statement causes an error to be signalled,
 638   0698  1  !       there is no RTL handler in the stack frame to catch the error
 639   0699  1  !       and clean up in the case of an unwind
 640   0700  1  !
 641   0701  1  !       This handler is enabled at the user's stack frame level.  The
 642   0702  1  !       address of whatever user handler that was in the frame is stored
 643   0703  1  !       in the ISB.  When an error is signalled, this handler finds
 644   0704  1  !       the address of the user handler, if any, and calls it.  There
 645   0705  1  !       should be no normally detectable difference caused by FOR$$IO_IN_PROG
 646   0706  1  !       being on the frame.  On unwind, the current ISB is popped and the
 647   0707  1  !       user's handler is called again.  This way, we are protected against
 648   0708  1  !       all errors on all call levels.
 649   0709  1  !
 650   0710  1  ! FORMAL PARAMETERS:
 651   0711  1  !
 652   0712  1  !       SIG_ARGS_ADR.ml.ra      Address of signal arguments list
 653   0713  1  !       MCH_ARGS_ADR.ml.ra      Address of mechanism arguments list
 654   0714  1  !
 655   0715  1  ! IMPLICIT INPUTS:
 656   0716  1  !
 657   0717  1  !       ISB/LUB/RAB database
 658   0718  1  !
 659   0719  1  ! IMPLICIT OUTPUTS:
 660   0720  1  !
 661   0721  1  !       ISB/LUB/RAB database
 662   0722  1  !
 663   0723  1  ! COMPLETION CODES:
 664   0724  1  !
 665   0725  1  !       Whatever is returned by the user handler.
 666   0726  1  !--
 667   0727  1
 668   0728  2     BEGIN
 669   0729  2
 670   0730  2     GLOBAL REGISTER
 671   0731  2         CCB = 11 : REF $FOR$CCB_DECL;
 672   0732  2
 673   0733  2     BUILTIN
 674   0734  2         CALLG,
 675   0735  2         AP;
 676   0736  2
 677   0737  2     LOCAL
 678   0738  2         USER_HANDLER,                                 ! Address of user's handler
 679   0739  2         EST_FP : REF BLOCK [, BYTE];                  ! Establisher's FP
 680   0740  2
 681   0741  2     MAP
```

```
:   682        0742  2              SIG_ARGS_ADR : REF BLOCK [, BYTE],          ! signal argument list
:   683        0743  2              MCH_ARGS_ADR : REF BLOCK [, BYTE];          ! mechanism argument list
:   684        0744  2
:   685        0745  2      !+
:   686        0746  2      ! Get establisher's FP
:   687        0747  2      !-
:   688        0748  2
:   689        0749  2      EST_FP = .MCH_ARGS_ADR [CHF$L_MCH_FRAME];
:   690        0750  2
:   691        0751  2      !+
:   692        0752  2      ! See if we are unwinding.
:   693        0753  2      !-
:   694        0754  2
:   695        0755  2      IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STS$V_COND_ID;, BYTE] EQL (SS$_UNWIND^-3)
:   696        0756  2      THEN
:   697        0757  2          BEGIN
:   698        0758  2          FOR$$CB_GET ();                              ! Get address of current LUB
:   699        0759  3
:   700        0760  3          IF .EST_FP NEQ .CCB [ISB$A_USER_FP] THEN FOR$$SIG_FATINT ();     ! Error
:   701        0761  3
:   702        0762  3          USER_HANDLER = .CCB [ISB$A_USR_HANDL];  ! Get user's handler address
:   703        0763  3          CLEANUP_LUB (FOR$K_UNWINDPOP);            ! Clean up LUB and restore user's handler
:   704        0764  3
:   705        0765  3          IF .USER_HANDLER NEQ 0 THEN RETURN CALLG (.AP, .USER_HANDLER);
:   706        0766  3
:   707        0767  3          RETURN SS$_NORMAL;
:   708        0768  3          END;
:   709        0769  2
:   710        0770  2      !+
:   711        0771  2      ! This is a signal.  Find the ISB that matched the establisher's
:   712        0772  2      ! FP.
:   713        0773  2      !-
:   714        0774  2
:   715        0775  2      FOR$$FP_MATCH (.EST_FP);
:   716        0776  2
:   717        0777  2      !+
:   718        0778  2      ! Call user's handler and return.
:   719        0779  2      !-
:   720        0780  2
:   721        0781  2      USER_HANDLER = .CCB [ISB$A_USR_HANDL];
:   722        0782  2
:   723        0783  2      IF .USER_HANDLER NEQ 0 THEN RETURN CALLG (.AP, .USER_HANDLER) ELSE RETURN SS$_RESIGNAL;
:   724        0784  2
:   725        0785  1      END;                                              ! End of FOR$$IO_IN_PROG
```

```
                                         080C 00000          .ENTRY   FOR$$IO_IN_PROG, Save R2,R3,R11        : 0685
                                  50      08  AC D0 00002     MOVL     MCH_ARGS_ADR, R0                       : 0749
                                  53      04  A0 D0 00006     MOVL     4(R0), EST_FP
                                  50      04  AC D0 0000A     MOVL     SIG_ARGS_ADR, R0
00000124  8F      04  A0         19      03  ED 0000E     CMPZV    #3, #25, -4(R0), #292                  : 0755
                                  28      12 00018         BNEQ     2$
                        00000000G 00      16 0001A         JSB      FOR$$CB_GET                            : 0758
                  FF4C  CB                53  D1 00020     CMPL     EST_FP, -180(CCB)                     : 0760
```

```
                                       07 13 00025                BEQL      1$
                  00000000G  00        00 FB 00027                CALLS     #0, FOR$$SIG_FATINT
                            52  FF44    CB D0 0002E  1$:          MOVL      -188(CCB), USER_HANDLER
                                       7E D4 00033                CLRL      -(SP)
                  0000V  CF             01 FB 00035                CALLS     #1, CLEANUP_LUB
                                       52 D5 0003A                TSTL      USER_HANDLER
                                       14 12 0003C                BNEQ      3$
                            50         01 D0 0003E                MOVL      #1, R0
                                       04 00041                   RET
                                       53 DD 00042  2$:           PUSHL     EST_FP
                  00000000G  00        01 FB 00044                CALLS     #1, FOR$$FP_MATCH
                            52  FF44    CB D0 0004B                MOVL      -188(CCB), USER_HANDLER
                                       04 13 00050                BEQL      4$
                            62         6C FA 00052  3$:           CALLG     (AP), (USER_HANDLER)
                                       04 00055                   RET
                            50  0918   8F 3C 00056  4$:           MOVZWL    #2328, R0
                                       04 0005B                   RET
```

; Routine Size: 92 bytes,   Routine Base: _FOR$CODE + 01DA

; 726   0786 1

```
728    0787   1  ROUTINE CLEANUP_LUB (ACTION) : NOVALUE =
729    0788   1
730    0789   1  !++
731    0790   1  ! FUNCTIONAL DESCRIPTION:
732    0791   1  !
733    0792   1  !      Perform the UNWIND action indicated by ACTION on the current LUB.
734    0793   1  !
735    0794   1  ! FORMAL PARAMETERS:
736    0795   1  !
737    0796   1  !      ACTION.rlu.v              FOR$K_UNWINDNOP, FOR$K_UNWINDPOP, or FOR$K_UNWINDRET.
738    0797   1  !
739    0798   1  !--
740    0799   1
741    0800   2     BEGIN
742    0801   2
743    0802   2     GLOBAL REGISTER
744    0803   2         CCB = 11 : REF $FOR$CCB_DECL;
745    0804   2
746    0805   2     BIND
747    0806   2         FAB = CCB: REF $FOR$FAB_CCB_STRUCT;
748    0807   2
749    0808   2     CASE .ACTION FROM FOR$K_UNWINDPOP TO FOR$K_UNWINDRET OF
750    0809   2         SET
751    0810   2
752    0811   2         !+
753    0812   2         ! If the UNWIND action is to pop the LUB/ISB/RAB, call CB_POP to do
754    0813   2         ! the work.
755    0814   2         !-
756    0815   2
757    0816   2         [FOR$K_UNWINDPOP] :
758    0817   3             BEGIN
759    0818   3
760    0819   3             LOCAL
761    0820   3                 USER_FP;                              ! User's FP
762    0821   3
763    0822   3             FOR$$CB_GET ();                           ! CCB set to adr. of current /LUB/ISB/RAB
764    0823   3             USER_FP = .CCB [ISB$A_USER_FP];           ! Get user's FP
765    0824   3
766    0825   3             IF .USER_FP NEQ 0 THEN .USER_FP = .CCB [ISB$A_USR_HANDL];    ! Restore user's handler
767    0826   3
768    0827   3             CCB [RAB$L_UBF] = .CCB [LUB$A_RBUF_ADR];
769    0828   3             CCB [RAB$W_USZ] = .CCB [LUB$W_RBUF_SIZE];
770    0829   3             FOR$$CB_POP ();
771    0830   2             END;
772    0831   2
773    0832   2         !+
774    0833   2         ! If the UNWIND action is NOP, do nothing.
775    0834   2         !-
776    0835   2
777    0836   2         [FOR$K_UNWINDNOP] :
778    0837   2             ;
779    0838   2
780    0839   2         !+
781    0840   2         ! If the UNWIND action is RET, then try to $CLOSE the file associated
782    0841   2         ! with this LUB/ISB/RAB.  Deallocate any dynamic storage associated
783    0842   2         ! with this LUB.  Return the LUB/ISB/RAB to free storage.
784    0843   2
```

```
785       0844   2              [FOR$K_UNWINDRET] :
786       0845   3                  BEGIN
787       0846   3                  FOR$$CB_GET ();                                  ! Set CCB to adr. of current LUB/ISB/RAB
788       0847   3          !+
789       0848   3          ! See if file is RMS opened.
790       0849   3          !-
791       0850   3
792       0851   4                  IF (.FAB [FAB$W_IFI] NEQ 0)
793       0852   3                  THEN
794       0853   3          !+
795       0854   3          ! Do an RMS Close of the file, and arrange to deallocate its LUB/ISB/RAB
796       0855   3          ! when all I/O to it is finished.  Normally, we are doing the only I/O
797       0856   3          ! to it.
798       0857   3          !-
799       0858   3                      FOR$$CLOSE_FILE ()
800       0859   3                  ELSE
801       0860   3          !+
802       0861   3          ! Even though the file is not open, we wish to deallocate the LUB, since
803       0862   3          ! this is the simplest way to reinitialize it if the user tries to use
804       0863   3          ! the logical unit number again, so tell OTS$$POP_CCB to deallocate it.
805       0864   3          !-
806       0865   3                      CCB [LUB$V_DEALLOC] = 1;
807       0866   3
808       0867   3          !+
809       0868   3          ! We are done with the logical unit.
810       0869   3          !-
811       0870   3                  FOR$$CB_POP ();
812       0871   2                  END;
813       0872   2              TES;
814       0873   2
815       0874   1          END;
```

```
                          0804 00000 CLEANUP_LUB:
                                                      .WORD      Save R2,R11                              : 0787
                      52 00000000G  00  9E 00002       MOVAB      FOR$$CB_GET, R2
              02      00          04  AC CF 00009       CASEL      ACTION, #0, #2                          : 0808
              0020    003A            0006   0000E 1$:  .WORD      2$-1$,-
                                                                   7$-1$,-
                                                                   4$-1$
                      62          16  00014 2$:         JSB        FOR$$CB_GET                             : 0822
              50      FF4C        CB  D0 00016          MOVL       -180(CCB), USER_FP                      : 0823
              05      13          0001B               BEQL       3$                                       : 0825
              60      FF44        CB  D0 0001D          MOVL       -188(CCB), (USER_FP)
      24      AB      EC      AB  D0  00022 3$:         MOVL       -20(CCB), 36(CCB)                       : 0827
      20      AB      D2      AB  B0  00027            MOVW       -46(CCB), 32(CCB)                        : 0828
                      14          11  0002C             BRB        6$                                      : 0829
                      62          16  0002E 4$:         JSB        FOR$$CB_GET                             : 0846
              46      AB          B5  00030             TSTW       70(FAB)                                 : 0851
                      09          13  00033             BEQL       5$
      00000000G  00           00  FB  00035             CALLS      #0, FOR$$CLOSE_FILE                     : 0858
                      04          11  0003C             BRB        6$
      FF      AB                  10  88  0003E 5$:      BISB2      #16, -1(FAB)                            : 0865
                  00000000G  00  16  00042 6$:          JSB        FOR$$CB_POP                             : 0870
```

```
                                             04 00048 7$:     RET                                              ; 0874

; Routine Size:  73 bytes,    Routine Base:  _FOR$CODE + 0236


;  816        0875  1
;  817        0876  1 END                                                  !End of module
;  818        0877  1
;  819        0878  0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
_FOR$CODE	639	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]STARLET.L32;1	9776	18	0	581	00:01.0
_255DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	190	26	52	00:00.6
_255DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

```
;    BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:FORERROR/OBJ=OBJ$:FORERROR MSRC$:FORERROR/UPDATE=(ENH$:FORERROR)

; Size:            639 code + 0 data bytes
; Run Time:         00:16.9
; Elapsed Time:     00:45.4
; Lines/CPU Min:     3120
; Lexemes/CPU-Min: 15828
; Memory Used:  119 pages
; Compilation Complete
```

FORFIND
LIS

FORFMTINT
LIS

FOREXITHA
LIS

FORENODEF
LIS

FORENCOMF
LIS

FORIDATE
LIS

FORENDFIL
LIS

FORERRSNS
LIS

FOREXIT
LIS

FORERROR
LIS

FORFMTCP
LIS

FORENCOMO
LIS

FORINIDES
LIS